Practical :5A

Aim : Demonstrate the working of feature construction by combining and spiliting feature to extract the information from the dataset and write the conclusion about survival status of different categories

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
import os
```

In [2]:
```
df = pd.read_csv("train.csv")[["Age","Pclass","SibSp","Parch","Survived"]]
```

In [3]:
```
df.head()
```

Out[3]:

	Age	Pclass	SibSp	Parch	Survived
0	22.0	3	1	0	0
1	38.0	1	1	0	1
2	26.0	3	0	0	1
3	35.0	1	1	0	1
4	35.0	3	0	0	0

In [4]:
```
df.tail()
```

Out[4]:

	Age	Pclass	SibSp	Parch	Survived
886	27.0	2	0	0	0
887	19.0	1	0	0	1
888	NaN	3	1	2	0
889	26.0	1	0	0	1
890	32.0	3	0	0	0

In [5]:
```
df.dropna(inplace=True)
```

In [6]:
```
df.head()
```

Out[6]:

	Age	Pclass	SibSp	Parch	Survived
0	22.0	3	1	0	0
1	38.0	1	1	0	1
2	26.0	3	0	0	1
3	35.0	1	1	0	1
4	35.0	3	0	0	0

In [7]:
```
df.tail()
```

Out[7]:

	Age	Pclass	SibSp	Parch	Survived
885	39.0	3	0	5	0
886	27.0	2	0	0	0
887	19.0	1	0	0	1
889	26.0	1	0	0	1
890	32.0	3	0	0	0

In [8]:
```
x = df.iloc[:,0:4]
y = df.iloc[:,-1]
```

In [9]:
```
x.head()
```

Out[9]:

	Age	Pclass	SibSp	Parch
0	22.0	3	1	0
1	38.0	1	1	0
2	26.0	3	0	0
3	35.0	1	1	0
4	35.0	3	0	0

In [10]:
```
y.head()
```

Out[10]:
```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

In [11]:
```
## CV just means cross validation. Its a way of using all of your available training data to inform your model, while also using
```

In [12]:
```
np.mean(cross_val_score(LogisticRegression(),x,y,scoring="accuracy",cv=20))
```

Out[12]:
```
0.6933333333333332
```

In [13]:
```
## Appling Feature Construction
```

In [14]:
```
x["Family_size"] = x["SibSp"] + x["Parch"] + 1
```

In [15]:
```
x.head()
```

Out[15]:

	Age	Pclass	SibSp	Parch	Family_size
0	22.0	3	1	0	2
1	38.0	1	1	0	2
2	26.0	3	0	0	1
3	35.0	1	1	0	2
4	35.0	3	0	0	1

In [16]:
```
def myfunc(num):
    if num ==1:
        #alone
        return 0
    elif num>1 and num<=4:
        # small family
        return 1
    else:
        # large family
        return 2
```

In [17]:
```
myfunc(4)
```

Out[17]:
```
1
```

In [18]:
```
x["Family_type"] = x["Family_size"].apply(myfunc)
```

In [19]:
```
x.head()
```

Out[19]:

	Age	Pclass	SibSp	Parch	Family_size	Family_type
0	22.0	3	1	0	2	1
1	38.0	1	1	0	2	1
2	26.0	3	0	0	1	0
3	35.0	1	1	0	2	1
4	35.0	3	0	0	1	0

the inplace parameter helps you decide how you want to affect the underlying data of the Pandas object.

In [20]:
```
x.drop(columns=["SibSp","Parch","Family_size"],inplace=True)
```

In [21]:
```
x.head()
```

Out[21]:

	Age	Pclass	Family_type
0	22.0	3	1
1	38.0	1	1
2	26.0	3	0
3	35.0	1	1
4	35.0	3	0

In [22]:
```
np.mean(cross_val_score(LogisticRegression(),x,y,scoring="accuracy",cv=20))
```

Out[22]:
```
0.7003174603174602
```

Feature Splitting

In [23]:
```
df = pd.read_csv("train.csv")
```

In [24]:
```
df.head()
```

Out[24]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [25]:
```
df["Name"]
```

Out[25]:
```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
                           ...
886                              Montvila, Rev. Juozas
887                       Graham, Miss. Margaret Edith
888           Johnston, Miss. Catherine Helen "Carrie"
889                              Behr, Mr. Karl Howell
890                                Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

In [29]:
```
df["Title"] = df["Name"].str.split(',', expand = True)[1].str.split('.',expand = True)[0]
```

In [32]:
```
df[["Title","Name"]]
```

Out[32]:

	Title	Name
0	Mr	Braund, Mr. Owen Harris
1	Mrs	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	Miss	Heikkinen, Miss. Laina
3	Mrs	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	Mr	Allen, Mr. William Henry
...	...	...
886	Rev	Montvila, Rev. Juozas
887	Miss	Graham, Miss. Margaret Edith
888	Miss	Johnston, Miss. Catherine Helen "Carrie"
889	Mr	Behr, Mr. Karl Howell
890	Mr	Dooley, Mr. Patrick

891 rows × 2 columns

In [45]:
```
df.groupby("Title").mean()["Survived"].sort_values(False)
```

```
C:\Users\User38\AppData\Local\Temp/ipykernel_16816/4065086800.py:1: FutureWarning: In a future version of pandas all arguments
of Series.sort_values will be keyword-only
  df.groupby("Title").mean()["Survived"].sort_values(False)
```

Out[45]:
```
Title
 Capt            0.000000
 Don             0.000000
 Jonkheer        0.000000
 Rev             0.000000
 Mr              0.156673
 Dr              0.428571
 Col             0.500000
 Major           0.500000
 Master          0.575000
 Miss            0.697802
 Mrs             0.792000
 Mme             1.000000
 Sir             1.000000
 Ms              1.000000
 Lady            1.000000
 Mlle            1.000000
 the Countess    1.000000
Name: Survived, dtype: float64
```

In [46]:
```
df["Is_Married"] = 0
df["Is_Married"].loc[df["Title"]=='Mrs']=1
```

```
C:\Users\User38\anaconda3\lib\site-packages\pandas\core\indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self._setitem_single_block(indexer, value, name)
```

In [47]:
```
df["Is_Married"]
```

Out[47]:
```
0      0
1      0
2      0
3      0
4      0
      ..
886    0
887    0
888    0
889    0
890    0
Name: Is_Married, Length: 891, dtype: int64
```

Conclusion :Using Feature construction by using combing spliting Man Women Married our not

In [ ]:

In [ ]: